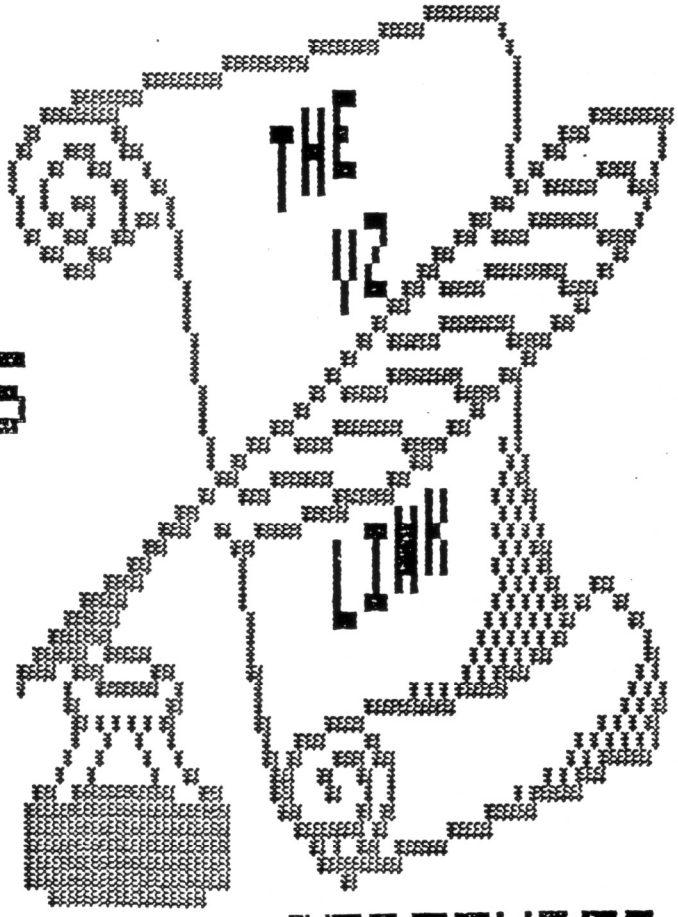
VZ-LINK MAGAZINE

NEW ZEALAND

VZ 300/200

COMPUTER USERS CLUB



ISSUE 35

SEPTEMBER 89

COMMENT

Hi folks,

Firstly,my apologies for the late newsletter.
Thank you once again to those that made the effort to attend the last meeting.
It is a shame that more members couldn't make it.
Maybe the next meeting in November will be the best yet.
It will be the last for the Year.The date will be advised next issue.
Once again I am forced to point out that producing a newsletter requires a constant source of material coming to hand for printing.
Any members who wish to contribute articles,programs listings etc.,are more than welcome to send their copy to me.
So please get that word processor (or pen) going to help me produce a lively,readable newsletter.
To all those regular contributors I say;

THANK YOU VERY MUCH FOR YOUR UNTIRING EFFORTS

from your hard working club organiser.

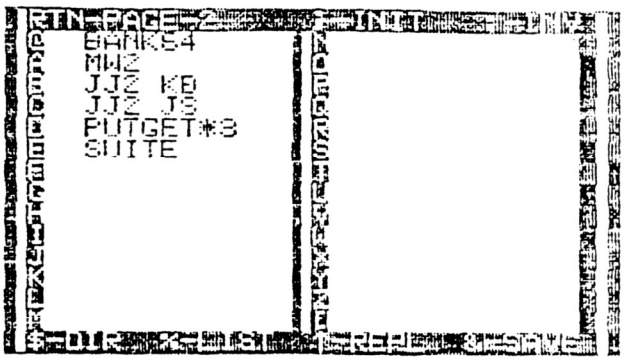
Bye for now

Peter Hill

TRIGONOMETRY

```
0 REM***TRIGONOMETRY***
1 '*(BY P. SHEPPARD)*
2 '**DO LINE 150 FIRST**
3 '*THEN POKE 31469,161*
4 '*150 PRINT X GO TO 160,170,180,190
5 CLS:MODE(0)
10 'INTRODUCTION
15 M=127:N=63
20 PRINTTAB(8);"TRIGONOMETRY":PRINT
30 FORI=1TO4:READD$(I):PRINTTAB(8);D$(I)
35 PRINT:NEXT:PRINT
40 PRINTTAB(2);"TYPE NUMBER CORRESPONDING"
50 PRINTTAB(6);"TO GRAPH WANTED"
60 I$=INKEY$
70 I$=INKEY$:X=VAL(I$):IFX<1ORX>4THEN70
80 'SET UP AXIS
90 MODE(1):COLOR2,1
100 FORH=0TO126:SET(H,32):NEXTH
105 FORV=0TO63:SET(0,V):NEXTV
110 RESET(63,32)      'INDICATES PI (180 DEG)
120 FORA=1TO126
130 '150 ON X GO TO 160,170,180,190
150 IFX=1GOTO160ELSEIFX=2GOTO170
155 IFX=3GOTO180ELSEIFX=4GOTO190
160 Y=33-25*SIN(A/20):GOTO175
170 Y=33-25*COS(A/20)
175 RESET(0,8):RESET(0,57):GOTO200
180 Y=33-1/SIN(A/20)/10:GOTO195
190 Y=33-1/COS(A/20)/10
195 RESET(0,29):RESET(0,36)
200 COLOR4:SET(A,Y):NEXTA
250 FORPAUSE=1TO1000:NEXT:RUN
300 DATA"1...Y=SIN(X)","2...Y=COS(X)"
310 DATA"3...Y=COSEC(X)","4...Y=SEC(X)"
```

PAGED DISK MENU BY P. & J.LEON



The two screen dumps above show two pages of the menu and top and bottom of frame show all relevant commands.

Lines 10 to 15 :-

DO NOT ALTER THEM - Type in exactly as shown. The (:) COLON in front of the command RUN is not a mistake. The program is designed to LOAD and RUN or BRUN basic or machine code programs.

In case of M/C programs the character 'B' is POKED into line 15 replacing the COLON and thereby changing the command RUN into BRUN. The program FILENAME is POKED also into line 15 where the the numbers 1 to 8 are between the quotes. For that reason any attempt to change line 10 and 15 will result in a corrupted program. In case there are less than 8 characters then line 205 puts blank spaces at end of filename to make up 8 characters.

MENU COMMANDS :-

%-LIST - Pressing SHIFT 5 will list lines 500 to 600 where the FILENAME and FILETYPE must be entered into DATA statements seperated by a comma. A 'T' denotes a basic file while a 'B' denotes a M/C file. and need not be typed in as your programs will have different names. Try MENUP,T as your first filename.

RTN-PAGE-1 - The RETURN key toggles between pages 1 and 5 provided each page holds filenames. The current page number is shown.

;-INIT - The (;) key will allow you to INITialise disks and has a QUIT option.

:-INV - This key toggles between NORMAL and INVERSE screens.

$-dir - This key displays the directory and a further function gives you printer printout option.

&-SAVE - This key will save the MENUP program to disk.

^-REPL - If your disk already has a MENUP program on it then SHIFT 'N' will first erase and then save it again using same filename. If you add, change or delete filenames from MENUP then SHIFT 'N' is used to update it.

NOTE - In line 110 "^" denotes SHIFT 'N', the UP ARROW key.

LODING PROGRAMS - With MENUP up and running simply press the required key to the left of the filename and your program will RUN or BRUN depending on type.

```
10 GOTO25
15 :RUN"12345678"
20 '"[illegible]
25 PA=0:DIMP$(140),F$(140)
30 CLS:IFPEEK(30744)<>96THENPOKE30744,96
35 POKE30862,80:POKE30863,52
40 N=N+1:READP$(N),F$(N):IFF$(N)<>"*"THENGOTO40ELSEN=N-1
45 :
50 CLS:COLOR7:POKE30777,1:IFPA*28+1>NTHENPA=0
55 PRINT"[illegible]"CHR$(241+PA)"[illegible]";
60 FOR I=1 TO 14
65 PRINT@(32*I),"[illegible]"CHR$(191+I)"  "P$(I+PA*28);
70 PRINTTAB(15)"[illegible]"CHR$(205+I)"  "P$(I+14+PA*28);TAB(30)"[illegible]";
75 NEXT
80 PRINT@480,"[illegible]";:POKE29183,229
85 POKE30777,35:GOSUB370
95 :
100 X$=INKEY$:X$=INKEY$:IFX$=""THEN100ELSEX=USR(X)
105 IFX$=":",POKE30744,ABS(PEEK(30744)=0)*96:SOUND20,1
110 IFX$="^"THEN250
115 IFX$="&"THEN260
120 IFX$="$"THEN300
125 IFX$=";"THEN400
130 IFX$="%"THENCLS:LIST500-
135 IFX$=CHR$(13)ANDN>28THENPA=PA+1:GOTO50
140 IFX$<"@"ORX$>"["THEN100
145 P=ASC(X$)-63+PA*28:IFP$(P)=""THEN100
150 CLS:PRINT@480,"[illegible] "F$(P)": "P$(P);
155 GOSUB370
195 :
200 IFF$(P)="B"THENPOKE31477,66ELSEPOKE31477,58
205 FORB=1TO8:POKE31479+B,ASC(MID$(P$(P)+"        ",B,1)):NEXT
210 GOTO15
240 :
250 CLS:PRINT@480,"[illegible] T: MENUP";:GOSUB370:ERA"MENUP"
260 CLS:PRINT@480,"[illegible] T: MENUP";:GOSUB370:SAVE"MENUP"
295 :
300 CLS:DIR:STATUS
305 PRINT@17,"[illegible]";:GOSUB370
310 R$=INKEY$:R$=INKEY$:IFINKEY$=""THEN310ELSEX=USR(X)
315 IFR$="Y"THEN340ELSEIFR$="N"THEN50ELSE310
340 IF INP(0)=254 OR INP(0)=246THEN350ELSE355
350 CLS:POKE30876,1:DIR:STATUS:LPRINT:RUN
355 PRINT@480," [illegible]";
360 GOSUB370:SOUND0,9;0,5:RUN
365 :
370 SOUND30,1;25,1:RETURN
375 :
400 PRINT@481," [illegible] ";:GOSUB370
405 I$=INKEY$:I$=INKEY$:IFI$=""THEN405ELSEX=USR(X)
410 IFI$=" "THEN415ELSEIFI$="Q"THEN80ELSE405
415 PRINT@481,"  [illegible]  ";:GOSUB370
420 INIT:GOTO300
495 :
500 DATA MENU,T,MENUP,T,CONVERT,T,HOUSE,T,M-WORDS1,T,CODE,T
505 DATA ELEMENTS,T,INKEY$,T,SIDEWAYS,T,BOLL.OPS,T,BAS/FIND,T
510 DATA TR/DEMO,T,LPRINT,T,SPELL,T,BAS/CLS,T,MATHS,T,NEWCLS,B
520 DATA KAL,T,FIND,B,BM&C,T,SOUNDS,T,MUSICBOX,T,INTER,T
530 DATACAR-EXP,T,FOOTY,T,GR-EDIT,T,HEX-CONS,T,FLASK,T,BANK64,T
560 DATA MW2,T,JJ2 KB,T,JJ2 JS,T,PUTGET*3,T,SUITE,T
600 DATA ,*
```

HI RES SCREEN MOVE

HI-RES SCREEN MOVE & PRESERVE UTILITIES.
by
Bob Kitch.

Four subroutines are provided that can be called from within any BASIC program. Often it is desirable to be able to preserve a 2K Hi-res screen so that a Lo-res (text) screen can be displayed. After certain instructions are carried out, the Hi-res screen may need to be restored. The BASIC commands SET & RESET are much too slow for this application. The fast Block Move command inherent in the Z80 is made for the task.

The subroutines given are general and relocatable for any VZ. The Block Move construct is called via the USR command. The Move routine is modified during execution so that it is used as a Block Fill routine and also moves the screen image both ways between Video RAM and the Hi-mem buffer.

1. SUBROUTINE 3000 - INITIALIZATION.

This section should be called early in the mainline program. It lowers TOM, reserves 2400 bytes of Hi-mem, loads 14 bytes of machine code and sets up a number of pointers. Ensure that these variables are not re-used during program execution.

MR & LR - MSB and LSB of starting address for Routine.
MS & LS - ditto for Storage buffer.
MV & LV - ditto for Video RAM.
SS & SS+1 - address for Source.
DS & DS+1 - address for Destination.
ZS & ZS+1 - address for LD (HL),XX bytes.

The 14 bytes of M/L are loaded into the buffer and are first set up as a Destructive Block Fill.

```
LD HL,SOURCE
LD DE,DESTINATION
LD BC,SIZE
LD (HL),CHARACTER
LDIR
RET
```

The initial colour of the screen buffer can be set by altering the second piece of data loaded in line 3330. The addresses for SOURCE and DESTINATION are initially set to dummy values of 255D.

2. SUBROUTINE 4000 - CLEAR HI-MEM BUFFER.

This is the Block Fill call which is also needed early in the program to fill the 2K buffer with a certain CHARACTER. It sets up the USR Vector, pokes in the SOURCE and DESTINATION addresses and finally modifies the M/L code to a Block Move type.

```
LD HL,BUFFER     ;LS,MS
LD DE,BUFFER+1   ;L1,M1
LD BC,2048D
NOP
NOP
LDIR
RET
```

3. SUBROUTINE 4000 - MOVE SCREEN TO HI-MEM BUFFER.

This routine is called whenever the video screen located from 7000H to 77FFH is to be moved into the Hi-mem buffer. A Lo-res screen can then be displayed under program control.

```
LD HL,7000H    ;LV,MV
LD DE,BUFFER   ;LS,MS
LD BC,2048D
NOP
NOP
LDIR
RET
```

4. SUBROUTINE 5000 - MOVE HI-MEM BUFFER TO VIDEO RAM.

This sets up the Block Move as shown below and restores the Hi-res screen that had previously been buffered.

```
LD HL,BUFFER
LD DE,7000H
LD BC,2048D
NOP
NOP
LDIR
RET
```

Well that's about it! Generally speaking, SUBROUTINES 3000 and 4000 should be run early in the calling program, as they set up and pre-load the buffer area. I trust that users find these routines very useful - they have been invaluable when writing graphics editors.

As a Footnote, it is worth recording the action of the LDIR opcode. It's action is as follows :-

i/ Assign (HL) to (DE)
ii/ INC HL
iii/ INC DE
iv/ DEC DE
v/ Repeat until BC=0

```
1 '****************************
2 '***  SCREEN MODE MOVER   ***
3 '*** A SET OF SUBROUTINES ***
4 '***   TO SWITCH BETWEEN  ***
5 '***      HI & LO RES     ***
6 '***           BY         ***
7 '***       BOB KITCH      ***
8 '***          6/87        ***
9 '****************************
10 '
15 'THIS SET OF SUBROUTINES IS USED IN BRUCE'S ART GALLERY.
```

```
2998 '
2999 '***INITIALIZATION - CALL AT START OF PROGRAM.
3000 M1=PEEK(30898):L1=PEEK(30897)    :'***FIND TOM.
3010 TM=M1*256+L1                     :'***TOM.
3020 TM=TM-2400                       :'***RESERVE 2.4K BUFFER.
3030 MS=INT(TM/256):LS=TM-MS*256      :'***NEW TOM.
3040 POKE 30898,MS:POKE 30897,LS      :'***SET NEW TOM.
3050 CLEAR 300                        :'***RESET POINTERS.
3060 TM=PEEK(30898)*256+PEEK(30898)   :'***NEW TOM.
3070 TM=TM+1                          :'***START ADDR. BUFFER.
3080 MR=INT(TM/256):LR=TM-MR*256      :'***START POINTERS.
3090 FOR MS=TM TO TM+13               :'***LOAD IN M/C.
3100    READ LS:ID=MS                 :'***SET UP ADDR.
3110    IF ID>32767 THEN ID=ID-65536 :'***CONV. TO S.INT.
3120    POKE ID,LS                    :'***POKE IN MOVE CODE.
3130 NEXT MS
3140 SS=TM+1                          :'***ADDR. SOURCE.
3150 IF SS>32767 THEN SS=SS-65536     :'***CONV. TO S.INT.
3160 DS=TM+4                          :'***ADDR. DESTIN.
3170 IF DS>32767 THEN DS=DS-65536     :'***CONV. TO S.INT.
3180 ZS=TM+9                          :'***ADDR. 2K BUFF.
3190 IF ZS>32767 THEN ZS=ZS-65536     :'***CONV. TO S.INT.
3200 TM=TM+30                         :'***START OF SCRN BUFF.
3210 MS=INT(TM/256):LS=TM-MS*256      :'***START OF 2K BUFF.
3220 MV=112:LV=0                      :'***START OF VIDEO RAM.
3230 RETURN
3298 '
3299 '***DESTRUCTIVE BLOCK FILL ROUTINE.
3300 DATA 033,255,255     :'LD HL,SOURCE.
3310 DATA 017,255,255     :'LD DE,DESTINATION
3320 DATA 001,000,008     :'LD BC,SIZE
3330 DATA 054,000         :'LD (HL),CHARACTER
3340 DATA 237,176         :'LDIR
3350 DATA 201             :'RET
3998 '
3999 '***CLEAR BUFFER ROUTINE.
4000 POKE 30863,MR:POKE 30862,LR      :'***SET USR() POINTERS.
4010 POKE SS,LS:POKE SS+1,MS          :'***SOURCE ADDR.
4020 M1=MS:L1=LS+1                    :'***CALC SOURCE+1.
4030 IF L1>255 THEN L1=0:M1=M1+1      :'***CHECK RANGE.
4040 POKE DS,L1:POKE DS+1,M1          :'***DESTINATION ADDR.
4050 S=USR(0)                         :'***SET 2K BUFFER.
4060 POKE ZS,0:POKE ZS+1,0            :'***INSERT NOP'S
4070 RETURN
4998 '
4999 '***MOVE 2K SCREEN BUFFER TO TOM BUFFER.
5000 POKE 30863,MR:POKE 30862,LR      :'***SET USR() POINTERS.
5010 POKE SS,LV:POKE SS+1,MV          :'***SET SOURCE ADDR.
5020 POKE DS,LS:POKE DS+1,MS          :'***SET DESTINATION ADDR.
5030 S=USR(0)                         :'***MOVE SCREEN TO HI-MEM.
5040 RETURN
5998 '
5999 '***MOVE TOM BUFFER INTO 2K VIDEO RAM...
6000 POKE 30863,MR:POKE 30862,LR      :'***SET USR() POINTERS.
6010 MODE(1)                          :'***HI-RES MODE.
6020 POKE SS,LS:POKE SS+1,MS          :'***SET SOURCE ADDR.
6030 POKE DS,LV:POKE DS+1,MV          :'***SET DESTINATION ADDR.
6040 S=USR(0)                         :'***MOVE SCREEN TO HI-MEM.
6050 RETURN
7000 END
```

MAKING C.W.P WORK BY JOE LEON

The Cartridge Word Processor will not work with the SUPER II VZ200 (34K RAM) or VZ300 18K Memory Expansion (INTERNAL). It will also not work if any RAM (2K) is active at 6000-67FF Hex. VZ owners with the MAGIC EPROM fitted should have no problems as they can switch it out easily if it does'nt work with C.W.P. unless they have extra RAM fitted like I have. So far I have'nt had time to modify my spare VZ with the Magic Eprom. The C.W.P. needs at least 12K of unused Address Range from D000 to FFFE Hex to become operational. This article explains how to free the address range needed.

SUPER II VZ200_MODIFICATIONS :-

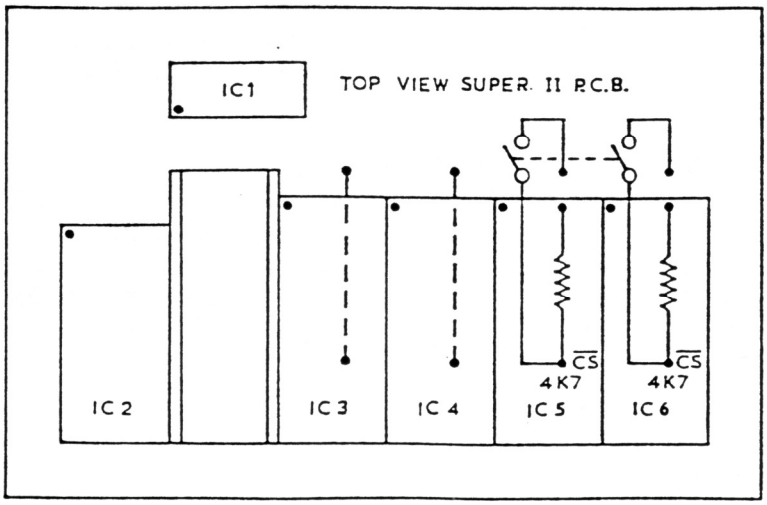
If you study the diagram you will realise the mod. is a simple one, just a couple of pull up resistors and a switch is all that's needed. The CS (Chip Select) line on IC 5 and 6 must be modified as shown. With the switch in the open position both IC 5 and 6 are deactivated allowing the C.W.P. to operate. Make sure the power is off when using the switch.

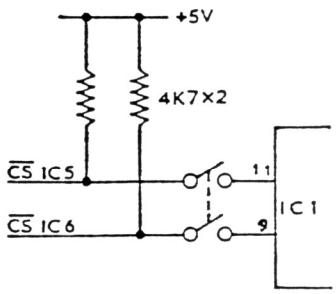
Remove IC 5 and 6 from the board. Next if you have sockets then the middle of them have to be cut out so you can remove the two Link Wires. Drill two small holes in the 5V track. The 5V track is easy to find. It runs under pins 1 and 28 of IC's 3 to 6. Fit the resistors and switches as shown. The switch can be a DPST, DPDT or a Dip Switch. In my case I used a four way Dip Switch which allows me to switch out IC'S 4 to 6 and Zero Power Ram at 6000-67FF Hex. One benefit of being able to switch out the Rams singly is that it allows me to test out VZ200/300 16K Ram Expansion cartridges. The alteration shoud'nt present any problems as it's a simple one, but do take care with the mods.

VZ owners with extra internal RAM get a bonus with the W.P.C. by way of extra memory for text. EG :-

VZ200 . . . . . = 5324 Bytes free for text.
VZ300 . . . . . = 15564 Bytes free for text.
VZ200 Super II - 24K = 9420 Bytes free for text.
VZ200 Super II - 16K = 17612 Bytes free for text.

If you have any Ram or Eprom at 6000-67FF Hex it will also have to be deactivated using the same procedure as with Super II.





VZ dos

After having looked at the directory and track map last edition, this month we're going to look deeper into how the DOS is managed, and present a couple of routines you can experiment with.

When you first switch on your VZ, equiped with disk system, the drive starts up, a few frightening sounds are heard, and the DOS title appears on the screen. Actually, a lot more has occured in that second or so. Firstly, the head of the drive has been moved to its outermost position (track#0). This is necessary since the DOS has no idea where the head was left the last time the drive was used, and the only way to be sure of its position is to move it up against the track 0 stopper. This ensures that the head is above track 0.

Since the DOS needs to keep track (no pun intended) of where it is on the disk, and many other variables, it needs some RAM it can use to hold such information. At switch on, the DOS 'steals' 310 bytes of RAM from the top of available memory. This area (sometimes called the DOS workspace or scratchpad) is then left untouched by the BASIC system, as the top of memory pointer is set just below it.

The 310 bytes of 'scratchpad' contain tables and data buffers. Firstly, we'll look at the DOS vector table. You'll find this in the VZ300 Technical Reference Manual, but there are a couple of errors, so we've reproduced it here. The DOS vector table is virtually a list of pieces of data relating to the operation of the DOS. Since its absolute location in memory will vary depending on how much memory you have, the VZ uses one of the registers in the Z80 to keep track of where it starts. The IY register is known as an index register since it is often used to handle tables such as the DOS vector table. Because the DOS uses the IY register, the programmer is both advantaged and disadvantaged. The advantage lies in the fact that it is very easy to read and write data to the table, from machine code, using special indexed instructions (see 'VZ ASSEMBLY' series). However, it is also a disadvantage since any machine code program which already uses the IY register must be modified so that its contents are preserved. Should the value stored in the IY register be lost, then the ODS will crash. Here is the table:

FUNCTION:	OFFSET:	BYTES:	NAME:
FILE#	IY+00	1	FILNO
FILENAME	IY+01	8	FNAM
FILE TYPE	IY+09	2	TYPE
DRIVE#	IY+11	1	DK
REQUEST CODE	IY+12	1	RQST
SOURCE DRIVE FOR DCOPY	IY+13	1	SOURCE
USER BUFFER ADDRESS PTR	IY+14	2	UBFR
DEST. DRIVE FOR DCOPY	IY+16	1	DESTIN
USER SPEC. SECTOR NUMBER	IY+17	1	SCTR
USER SPEC. TRACK NUMBER	IY+18	1	TRCK
RETRY COUNT	IY+19	1	RETRY

CURRENT TRACK NUMBER	IY+20	1	DTRCK
NEXT SECTOR NUMBER	IY+21	1	NSCT
NEXT TRACK NUMBER	IY+22	1	NTRK
FILE CONTROL BLOCK 1	IY+23	13	FCB1
FILE CONTROL BLOCK 2	IY+36	13	FCB2
DATA BUFFER ADDRESS PTR	IY+49	2	DBFR
COPY OF LATCH	IY+51	1	LTHCPY
TRACK/SECTOR MAP ADDRESS	IY+52	2	MAPADR
TRACK COUNT FOR DCOPY	IY+54	1	TRKCNT
TRACK POINTER FOR DCOPY	IY+55	1	TRKPTR
STEPPER MOTOR PHASE	IY+56	1	PHASE

Each of the DOS vectors is given an offset, expressed as IY + something. This simply gives the address of that particular vector. For example, the retry count can be found at the location equal to the contents of the IY register plus 19 (IY+19). The 'Bytes' column indicates how many bytes each vector uses. Notice that the filename is 8 bytes long, as we would expect. The 'Name' column gives an abbreviated name to each functiom. These are of little importance; they are only used by the Technical Manual.

Don't worry about understanding what all the vectors mean at this stage. We'll be explaining each as we come to it.

Let's take a break from the theory and try some practical examples. Following is a BASIC program which we will be building on each edition. It allows the user to read or write to any sector on the disk. Its operation is very similar to the Sector Editor commercially available (Laserlink's Disk Doctor), but it doesn't allow the convenience of on-screen editing (you may like to work on this yourself).

The program uses a short machine code routine, the source code of which is listed below. This is used to access other machine code routines in the VZ DOS, which we will examine later. Don't worry about trying to understand the operation of this routine; suffice to say that it reads or writes a sector to the disk. The routine is POKEd into screen RAM by the basic program.

The section of the program listed here contains just the start-ups and sector read option. We'll add the writing option next time. Here are the listings:

ASSEMBLY LANGUAGE SOURCE CODE LISTING

```
001     PUSH AF
002     PUSH BC
003     PUSH DE
004     PUSH HL
005     DI
006     CALL 4008H
007     CALL 4035H
008     OR   A
```

```
009        JP   NZ,400EH
010        CALL 400BH
011        EI
012        POP  HL
013        POP  DE
014        POP  BC
015        POP  AF
016        RET
```

BASIC PROGRAM LISTING

```
5 IY=PEEK(30897)+256*PEEK(30898)+1-65536
6 DIM B(128): POKE 30862,0: POKE30863,114
8 GOSUB 1000
10 CLS:PRINT"    SECTOR READ/WRITE UTILITY"
20 PRINT:PRINT:INPUT"READ OR WRITE SECTOR (R/W)";Z$
30 IF Z$="R" THEN POKE 29193,53 ELSE POKE 29193,50
40 PRINT:INPUT"TRACK #";T: INPUT"SECTOR #";S
50 POKE IY+18,T: POKE IY+17,S
60 IF Z$="W" THEN 500
70 X=USR(0)
80 BUFF=PEEK(IY+49)+256*PEEK(IY+50)
90 FOR Z=1 TO 128
100 B(Z)=PEEK(Z+BUFF-65537)
110 NEXT Z
120 CLS: FOR Z= 1 TO 128
130 H=INT(B(Z)/16)
140 IF H>9 THEN H$=CHR$(55+H) ELSE H$=CHR$(48+H)
150 L=B(Z)-H*16
160 IF L>9 THEN H$=H$+CHR$(55+L) ELSE H$=H$+CHR$(48+L)
165 P=INT((Z-1)/8) :PRINT@ P*32+(Z-P*8-1)*3,H$
167 IF B(Z)>31 AND B(Z)<123 THEN H$=CHR$(B(Z)) ELSE H$="."
170 PRINT@ P*24+23+Z,H$
180 NEXT Z
190 GOTO 190
1000 FOR Z=29184 TO 29207: READ A: POKE Z,A: NEXT : RETURN
1010 DATA 245,197,213,229,243,205,8,64,205,53,64,183,194,14,64
1020 DATA 205,11,64,251,225,209,193,241,201
65324
```

Line 5 shows a simple method of obtaining the contents of the IY register (the start of the vector table). Firstly, the value of the top of memory pointer is obtained, then this value is incremented. 65536 is subtracted from the address so that we can use the IY variable in POKE statements.

Line 6 sets up the pointer for the machine code routine which starts at 29184 (7200H), and an array B which will store the 128 bytes read or written to a sector. Lines 40 and 50 ask for the track and sector number to be read/written to, and POKEs these into the relevant vectors, as shown in the table. Line 70 calls the machine code routine, which reads the sector into a data buffer, the location of

which we must obtain from another vector. Line 80 gets its address and stores it in a variable BUFF. The following loop reads the 128 bytes from the buffer into the B array.

Lines 120-180 convert each byte into a hex number, and display these across and down the screen. The ASCII representation is given where appropriate on the right hand side of the screen. Lines 1000-1020 contain the data for the machine code routine. Note the loop at line 190 - this holts the program.

Type in the program, checking it carefully when finished. Save it, and then RUN it. Place an initialised but valueless disk in the drive. Set track#0, sector#0. The drive should spin for a short time, followed by a delay, and then data should begin to appear. Each screen line contains eight hex numbers (16x8=128 bytes per sector) and their ASCII equivalents on the right hand side. Where a byte is not a printable character, a full stop is printed.

The screen will scroll when the bottom line is printed (can anyone think of a way round this without POKEing the screen characters?). You should recognise the directory entries in the data. Look back at the previous edition to determine what all the bytes following the filename represent.

Experiment with the program as much as you wish, but remember that the write function is not yet fitted. We'll cover this and some other DOS routines next month.

P.O.BOX 1972 C.P.O AUCKLAND N.Z

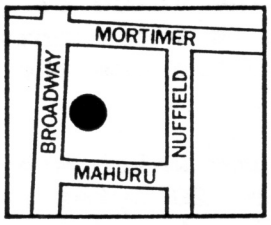
(C) AUCKLAND VZ 300/200 COMPUTER USER'S CLUB 1986

ALL MATERIAL IN THIS NEWSLETTER IS SUBJECT TO COPYRIGHT..
CONTRIBUTED PROGRAMS & ARTICLES CAN BE REPRODUCED ON THE UNDERSTANDING THAT THEY ARE FOR THE PRIVATE USE OF SUBSCRIBERS ONLY.
COPYRIGHT IS ALWAY'S RETAINED BY THE AUTHOR.
THIS CLUB WILL NOT BE HELD RESPONSIBLE FOR THE BREECHING OF COPYRIGHT CAUSED BY THE PUBLICATION OF ANY MATERIAL SENT IN UNDER A FALSE NAME.

YOU ARE WELCOME TO PHONE ME IF YOU HAVE PROBLEMS
PHONE 761-451

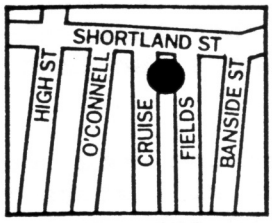
There is a Flash On Time Printer near you!

AUCKLAND



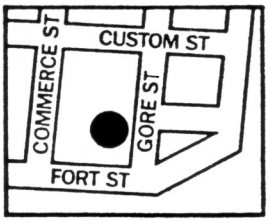
430 Broadway
Newmarket, Auckland
Tel: 520 3465
Fax: 520 5380

AUCKLAND



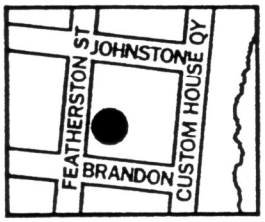
Level 3, Shortland Towers
Shortland Street
Auckland 1
Tel: 395 668
Fax: 392 910

AUCKLAND



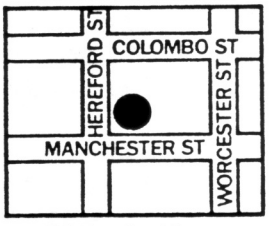
9 Gore Street
Auckland City
Tel: 395 888
Fax: 375 965

WELLINGTON



139 Featherston Street
P.O. Box 3808
Wellington 1
Tel: 730 446
Fax: 723 967

CHRISTCHURCH



167 Hereford Street
Christchurch 1
Tel: 661 723
Fax: 667 732

Look for our new-look at these locations...

Flash
The On Time Printer.

NEWSLETTER INDEX
ISSUES 27-33

ISSUE 27

PROGRAM-SCREENDRAW (BROOKE SPONG)
PROGRAM-CROSSWORDS (JOHN HOPE)
PRINTER INTERFACES (LAURIE WHETTON)
PROGRAM-PIZZA DELIVERY CO (KEITH WRIGHT)
PROGRAM-ZAP A DOT (KEITH WRIGHT)
REVIEW-32K RAM-64K EPROM (JOE LEON)

ISSUE 28

STRUCTURED PROGRAMMING PART 1 (CHRIS HOBROUGH)
PERIPHERALS FOR THE VZ (P RICHARDSON)
SUITE (ROBERT QUINN)
HEAD CLEANING DISK UPDATE
BIT BY BIT-Frequency Counter (BRUCE HARRIS)

ISSUE 29

PROGRAM-COMPUTOR BET (MARC ATKINSON)
SERIAL INTERFACES AND MODEMS
STRUCTURED PROGRAMMING PART 2 (CHRIS HOBROUGH)
DRIVE PROBLEMS (JOE LEON)
DRIVE EXT. LEAD (JOE LEON)
BIT BY BIT-SCREEN SCROLL

ISSUE 30

REVIEW-"LEAR JET"
PROGRAM-MONSTER HUNT PART 1 (KEITH WRIGHT)
STRUCTURED PROGRAMMING PART 3 (CHRIS HOBROUGH)
DISK DRIVE PROBLEMS (JOE LEON)
DOS PROBLEMS (JOE LEON)
PROGRAM-% ODDS IN LOTTO (KEN CLARKE)
VZ BASIC PROGRAM STORAGE & LINE RENUMBERING (GRAHAM MARSDEN)

ISSUE 31

PROGRAM-MONSTER HUNT PART 2 (KEITH WRIGHT)
BACKUP-DISK TO TAPE BINARY COPIER
MORE VZ ROM ROUTINES (DANIEL AYERS)
DO'S AND DON'TS OF COMPUTER CARE
REVIEW-VZ ASTERIODS
CUTTING THE MARGIN (L.CLARKE & A.R HILL)

ISSUE 32

BLOCK TRANSFERS FROM BASIC (CHRIS HOBROUGH)
SPLIT SCREEN-Mixed Graphics/Text
INTERRUPTS & Vectors PART 1 (BOB KITCH)
PROGRAM-DISK LABELS (DAVE BOYCE)
HOW TO USE VDTs AND STAY HEALTHY

ISSUE 33

VZ MAILING LIST UPDATE (KEN CLARKE)
INTERRUPTS & Vectors (BOB KITCH) PART 2

PROGRAM-REAL TIME CLOCK (BOB KITCH)
CUSTOMIZING W.P (DAVE BOYCE)
VZ DOS PART 1
NEWCOMERS START HERE